Vol.2018-SE-200 No.5
2018/12/3

変更要求に対するシステム設計書修正箇所抽出法の検証

髙橋加寿子 †1　塚本良太 †1　磯田誠 †1

概要：ソフトウェアの派生開発では、利用するソフトウェア成果物の変更による影響分析が重要であり、変更要求に基づき設計書から適切な変更箇所を抽出し変更する必要がある。しかし、変更箇所の特定は開発者の能力に依存するため、精度や品質が異なるという問題がある。そこで、変更要求から変更箇所の候補を抽出する手法を検証した。

キーワード：派生開発，変更要求，

Verification of the method of associating change request with system design document

KAZUKO TAKAHASHI†1　RYOTA TSUKAMOTO†1
MAKOTO ISODA†1

1. はじめに

1.1 背景

ソフトウェア開発には、新規開発や既存資産を再利用する開発などが存在する。納期や開発規模の増大等の理由から新規開発が行われることは少ない。既存資産を再利用する開発には、プロダクトライン開発や派生開発があるが、プロダクトライン開発は再利用するコア資産の構築や管理方法に課題があるため､派生開発が用いられることが多い。そのため、現在のソフトウェア開発では派生開発が主流となっている。派生開発の上流工程では、後工程の工数見積りや最終的な製品品質等に大きく影響する重要な作業である、変更要求に対する影響分析を行っている。影響分析をする上で、設計書やソースコード等の再利用する対象のトレーサビリティ情報が正確に管理されている場合は、この情報を基に影響範囲を特定することで、抜け漏れなく分析することが可能である。しかし、設計書のように文章で記載されているものは、トレーサビリティ情報の管理が難しいため、影響範囲の特定は開発者の読解力や経験に依存する。そのため、開発者の能力によって、影響分析の精度・品質が異なるという課題がある。

1.2 目的

開発者の能力によって影響分析の精度･品質が異なるという課題を解決するため、異なる種類の変更要求に対する影響分析において、最適な手法を検討し、属人性・変更箇所抽出時間を削減する影響範囲の抽出法を提案する。

本稿では、変更要求の種類として、既存システムがサポートするツールの追加を対象とした。この変更要求の場合、既存ツールの記載箇所に変更を加えることが想定されるため、既存ツールからキーワードを取得し、変更箇所候補を抽出する。その際に、変更箇所候補として提示する範囲によって抽出の精度が異なるため、実文書を用いた変更箇所候補の抽出法の検証と、範囲による精度の違い調査を行った。

1.3 構成

本稿では、2 章で変更要求と影響分析を説明する題材として派生開発を、3 章で本稿にて検討している課題を述べる。4 章で課題に対する解決策を示し、解決策の検証結果を 5 章で述べる。6 章で本研究の結論を示す。

2. 派生開発

本章では、影響分析を重要視する開発である派生開発を説明する。

2.1 概要

派生開発とは、既に稼動しているシステムに対し、設計書やソースコードに変更を加えることで、新規機能の追加や既存の機能・操作性の改善を行うシステム開発手法である。派生開発の特徴として以下が挙げられる。

- 開発者は既存のシステムの作成者ではないことが多い。
- 変更の規模が小さい場合、1 人で開発を行う場合がある。
- 時間的・技術的に、既存システムの仕様を全て理解してから開発に取り掛かることが出来ない。

†1 三菱電機株式会社
Mitsubishi Electric Corporation.

2.2 プロセス

派生開発の開発プロセスと、各プロセスにおける開発者に求められる処理や能力を以下に示す。

- プロセス 1：変更要求の具体化
  - 既存システムに対する変更内容を具体化する。変更要求として直接提示されない非機能要求においても具体的な変更内容に落とし込む。
- プロセス 2：既存システムの仕様理解
  - 派生開発を行う開発者が既存システムに対する知見がない場合、設計書・ソースコードからシステムを理解する。
- プロセス 3：変更箇所抽出
  - 変更要求の具体化内容を反映するため、既存システムの設計書から変更するべき箇所を抽出する。
- プロセス 4：変更方法検討
  - 設計書から抽出した変更箇所に、どういった変更を加えるかを検討し、設計書を修正する。変更したことにより、他の部分に影響がないことを確認する。
- プロセス 5：S/W 設計
  - 設計書の内容を基にソースコードの変更方法を決定する。
- プロセス 6：ソースコード変更
  - S/W 設計で決定した内容をソースコードに反映する。
- プロセス 7：テスト
  - システムが仕様通りに実行されるかをテストする。
- プロセス 8：運用
  - 既存システムと修正したシステムを入れ替え運用する。

3. 課題

本節では、派生開発において発生する課題のうち本稿で検討する課題を示す。

3.1 変更箇所特定の精度・品質に属人性が生じる

派生開発では、新規開発と比較して実際の記述量(修正量)が少ないため、開発期間を短く設定されやすい。そのため、本来であれば各プロセスで 2.2 節のような様々な処理を必要とするが、プロセス 2, 3 を省略し、既存システムの仕様を完全に理解しないまま、設計書やソースコードに変更を加えるといった対応をする場合がある。このような設計書変更を行う場合、開発者は各設計書を確認しながら変更箇所を決定する。しかし、既存システムに対する知見がない場合は、設計書から必要な箇所を判断するために時間を要するのに加え、既存システムの理解度により変更箇所の抽出精度が異なる。一方、開発者が既存システムに対し知見がある場合でも、変更要求からある程度変更箇所を推定することが可能なため、設計書の確認不足による変更漏れが発生する可能性がある。よって、変更要求に基づき既存の設計書を変更するプロセスにおいて、開発者の能力によって、変更箇所抽出の精度や品質が安定しないという課題がある。

その結果、テスト・運用工程で不具合が発覚した場合、手戻り等の無駄な作業が発生する。加えて、設計書に変更内容を記載しない・記載が少なければ、設計書とソースコードの内容に齟齬が生じている状態となり、システムの保守や次回以降の派生開発が困難になる。

3.2 変更後のレビュー漏れ

派生開発では、既に使用されているシステムに対する変更のため、変更を行った箇所以外は正常に動作すると考えられやすい。そのため、プロセス 3,4 における設計書の変更後のレビューでは変更を行った箇所を重点的に確認される。特に、既存の設計書の量が多ければ、変更箇所以外は確認されない場合もある。そのため、変更箇所以外の場所に変更漏れがあった場合は、レビューにて検知されにくく、テストや運用時に発覚するという課題がある。

4. 解決策

3 章の課題を解決するため、変更要求から変更箇所候補を抽出する方法を検討した。最適な変更箇所候補の抽出法は変更要求の種類に応じて異なると仮定し、4.1 節において変更要求と抽出法の概要を示す。

4.1 節の内容を基に、3.1 節の課題に対する解決策として、開発者の能力に依存せず変更要求から設計書内の変更箇所を抽出する方法を 4.2.1 節に示す。3.2 節の課題に対する解決策として、変更要求と変更内容から変更漏れ・誤りを検出する方法を 4.2.2 節に示す。

4.1 変更要求の種類とそれに伴う抽出法

本節では抽出法と、それに対応する変更要求の種類や課題を示す。

4.1.1 キーワード検索

変更要求から適切なキーワードを決定し、設計書内のキーワードが含まれている箇所から変更箇所候補を抽出する。既存のシステムがサポートしているツールや言語のバリエーションを追加するといった変更要求では、既存のツールや言語の記述箇所に修正を行う。そのため、既存の記述箇所を抽出するキーワード検索が効果的である。課題として、

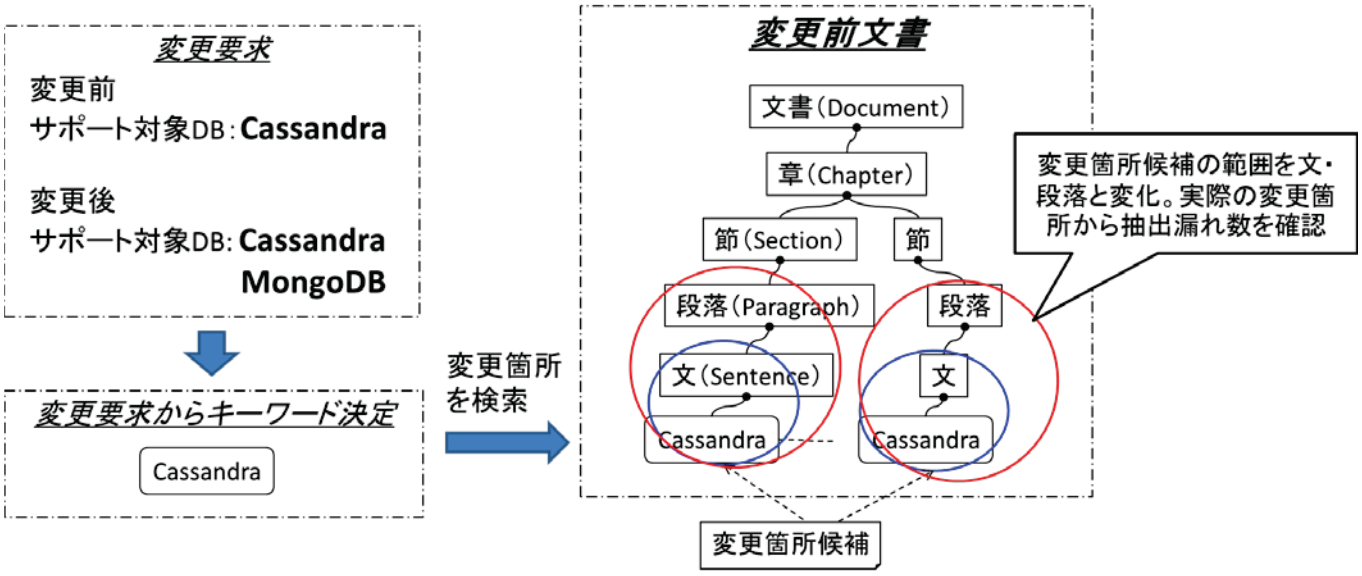

図 1　変更箇所候補の抽出

変更箇所候補の抽出精度はキーワードの決定方法によるところが大きく、最適なキーワードを検討する必要があることが挙げられる。また、システム固有名詞以外のキーワードを用いる場合は誤検出が発生することが想定される。

4.1.2 暗黙的参照

予め定義された設計書の章構成に着目し、関連する章・節から変更箇所候補を抽出する。新規サービス・システムの追加といった変更要求では、機能名やシステム構成などを追加する。そのため、必要な章・節を提示する暗黙的参照が効果的である。課題として、設計書の章構成が標準化されていることが前提となるため、独自の章構成を用いているものに対しては利用が困難となることが挙げられる。利用する章構成は、企業やプロジェクトごと決められたテンプレートが想定される。

4.1.3 明示的参照

設計書の作成者が文書内に記述したリンク(図表番号や章節番号等の参照)により設計書内の参照関係を抽出し変更箇所候補とする。課題として、設計書内のリンクの貼り方により抽出精度が異なることが挙げられる。

4.2　キーワード検索を用いた検証方法

本稿では、変更箇所候補の抽出と変更漏れ・誤りの検出法として 4.1.1 のキーワード検索を用いた方法を定義する。

4.2.1 変更箇所候補の抽出法

キーワード検索により変更箇所候補とその範囲を決定する方法を示す。キーワード検索が有効である変更要求として、システムで利用するツールなどのシステム固有名詞に関連する記述の追加・変更が挙げられる。この時、対象のシステム固有名詞が 1 つの場合は、該当するシステム固有名詞をキーワードとして使用する。対象のシステム固有名詞が複数あり、かつシステム固有名詞に関連がある場合は、各システム固有名詞に加えて、包括的に表現する上位概念等もキーワードとする。(例として、システム固有名詞がデータベース名の場合、データベース名として Cassandra や MongoDB などが挙げられる。このとき、各データベースをまとめて表現する場合の上位概念として NoSQL という記述も用いられることが想定される。)

キーワード検索により変更箇所候補を抽出する手順を示す。図 1 は変更要求の例としてサポート対象データベースの追加を示している。変更前のサポート対象データベースは Cassandra であり、変更後のサポート対象データベースは Cassandra と MongoDB となる。

1. キーワードを決定する。
   - 変更要求の内容からシステム固有名詞を抽出しキーワードを決定する。
2. キーワードの記載箇所を抽出する。
   - 変更前設計書内のキーワードが記載されている箇所を抽出する。

上記の手法において、変更箇所候補として提示する範囲を検証する必要がある。変更箇所候補の決定や変更漏れ・誤りを判定する場合、1 つの変更箇所候補として提示する範囲の違いにより、変更箇所候補や変更漏れ・誤りの抽出精度が異なる(図 2)。変更箇所候補の範囲を狭くするほど、確認すべき文字数が減るため修正に必要な時間は短くなるが、変更箇所抽出の精度も低くなり抽出漏れの可能性が高くなる。よって、最適な変更箇所候補の範囲を判定するため、変更箇所候補の範囲と抽出精度のバランスの検討も行う必

要がある。

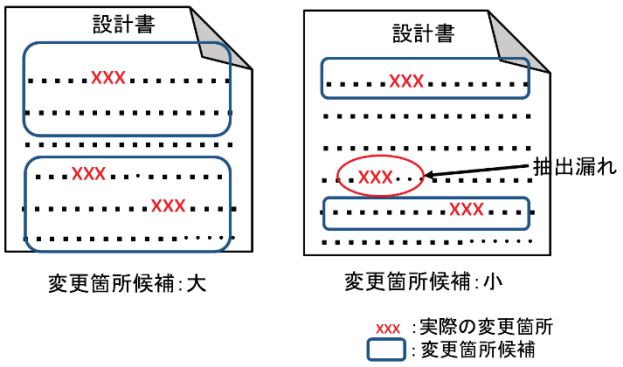

図 2　変更箇所候補の範囲

変更箇所候補の範囲を検証するため、上記の手順に加え、以下を実施する。

3. 変更箇所の抽出範囲に応じて、変更箇所候補を検証する。
   - ➢ キーワード検索において抽出したキーワードを含む「文」を変更箇所候補とした場合において以下を調査する
     - ✧ 変更箇所候補のうち、実際の変更箇所を含んでいる件数
     - ✧ 変更箇所候補以外の実際の変更箇所の件数
     - ✧ 変更箇所候補として抽出する範囲の割合

同様に、キーワードの記述を含む範囲を「文」から「段落」「節」・・・と広げ、変更箇所候補内に全ての変更箇所が含む範囲を調査する。

4.2.2 変更内容の判定方法

変更前設計書と変更後設計書を比較しキーワードの有無を確認することで、変更漏れや変更誤りの候補を判断する。

キーワード検索により変更内容の判定する手順を図 3 の変更要求を用いて示す。図 3 は変更要求の例としてサポート対象データベースの追加を示している。変更前のサポート対象データベースは Cassandra であり、変更後のサポート対象データベースは Cassandra と MongoDB となる。

以下の手順は、最適な変更漏れ・誤りの抽出範囲を判定するため、判定範囲を文・段落と変更し精度の検討を行う。

1. 変更前設計書と変更後設計書を比較する。
   - ● 変更前設計書に Cassandra の記述がある文・段落…において、変更後設計書に Cassandra と MongoDB の記載がある箇所は正しく変更されていると判定する。
   - ● 変更前設計書に Cassandra の記述がある文・段落…において、変更後設計書に変更されていない箇所は変更漏れの可能性があると判定する。
   - ● 変更前設計書に Cassandra の記述がある文・段

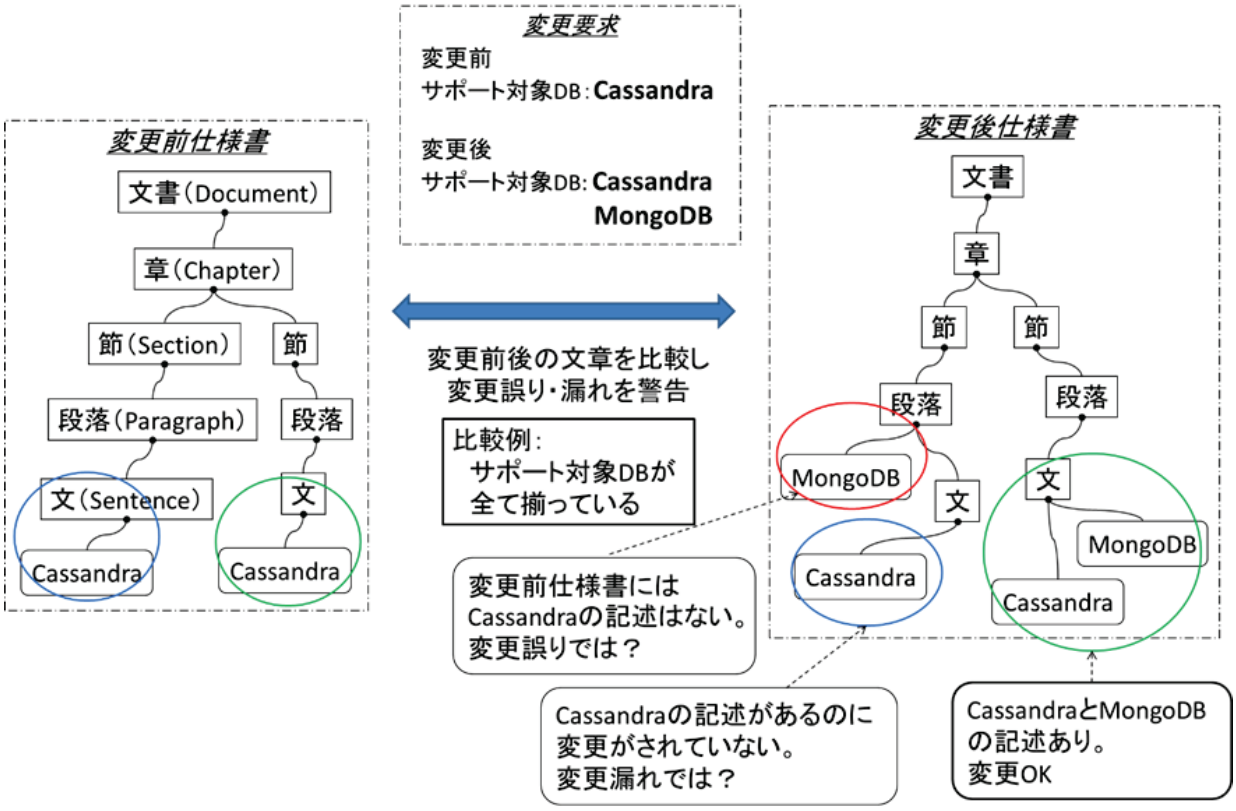

図 3　変更内容の判定

落…において、変更後設計書に Cassandra の記述がなく MongoDB の記述がある場合は変更誤りの可能性があると判定する。

- 変更前設計書の文・段落…において Cassandra の記述が無く、変更後設計書に MongoDB や Cassandra の記述がある箇所は変更誤りの可能性があると判定する。

2. 変更誤りに除外キーワードを設定する。
   - 変更誤りとされた箇所において、データベースの記述を MongoDB 以外の単語で表す場合がある。本検証対象では 2 つのデータベースをまとめて NoSQL と表記するため、変更後設計書に MongoDB の記述が無くとも NoSQL の記述がある場合は正しく変更されていると判断する。

5. 検証結果

実際に派生開発により変更を加える前と後の文書を、変更前設計書・変更後設計書とし、キーワード検索による抽出精度を検証した。

5.1 検証対象

検証に使用した設計書の概要を表 1 に示す。

表 1 検証対象設計書

	変更前設計書	変更後設計書
ページ数	24 ページ	25 ページ
文字数	12754 文字	12973 文字
変更箇所		34 文

変更前設計書に対する変更要求は、Cassandra のみであったサポート対象のデータベースに MongoDB を追加することである。

5.2 変更箇所抽出の検証結果

変更前設計書に対し 4.2.1 節の手順を実施し、変更箇所候補を抽出した。変更前設計書と変更後設計書の差分から実際に変更した箇所を抽出し、その場所と変更箇所候補を比較することで、キーワード検索による抽出率を検証した。

変更箇所候補として抽出した結果を表 2 に記載する。

抽出範囲を「文」とした場合、抽出した全ての変更箇所候補に実際の変更箇所を含んでいる。しかし、変更箇所候補以外にも 2 文実際の変更箇所があり、抽出漏れがあることがわかった。

抽出範囲を「段落」とした場合、全ての変更箇所候補に実際の変更箇所を含んでおり、抽出漏れもなく全ての変更箇所を抽出できることがわかった。段落では 1 つの段落に複数の変更箇所を含んでいる場合があるため、変更箇所候補数と変更数が文と異なる値を示している。

また、変更箇所候補の抽出量は、文の場合は設計書の 8.21%、段落の場合は 28.41%である。

表 2 変更箇所抽出結果

	抽出結果(文)	抽出結果(段落)
変更箇所候補数	28 文 (1047 文字)	14 段落 (3623 文字)
変更箇所候補割合	8.21%	28.41%
実際の変更数	30 文	14 段落
抽出漏れ	2 文	0

5.3 変更内容判定の検証結果

変更前・後設計書に対し、変更内容の判定方法 4.2.2 節の手順を実施することで、変更後設計書から変更漏れや変更誤りを検出する。

変更後設計書から変更内容を文単位で判定した結果を表 3 に示す。

変更後設計書に対して、キーワードとして NoSQL を設定しない場合は実際の変更箇所 34 件中 15 件を変更誤り・漏れと判定した。キーワードに NoSQL を設定した場合は変更誤り変更箇所 34 件中 8 件を変更誤り・漏れと判定した。

表 3 変更内容判定(抽出範囲：文)

変更前設計書記載内容	変更後設計書記載内容	検出数	判定
Cassandra	Cassandra/MongoDB	15	誤りなし
Cassandra	NoSQL	7	誤りなし(キーワード未設定時は変更漏れ)
Cassandra	Cassandra ～ MongoDB	4	誤りなし
Cassandra	Cassandra	2	変更漏れ
(DB 名記載無し)	MongoDB	2	変更誤り
(DB 名記載無し)	Cassandra/MongoDB	2	変更誤り
Cassandra	(DB 名記載無し)	2	変更誤り

変更後設計書から変更内容を段落単位で判定した結果を表 4 に示す。

変更後設計書に対して、キーワードとして NoSQL を設

定しない場合は実際の変更箇所 34 件中 2 件を変更漏れと判定した。キーワードに NoSQL を設定した場合は変更誤り・漏れの件数は 0 件と判定した。

表 4　変更内容判定(抽出範囲：段落)

段落番号	変更前設計書記載内容	変更後設計書記載内容	判定
1	Cassandra	Cassandra/MongoDB	誤りなし
	Cassandra	Cassandra/MongoDB	
	Cassandra	Cassandra/MongoDB	
	Cassandra	Cassandra/MongoDB	
2	Cassandra	NoSQL	誤りなし
	Cassandra	Cassandra/MongoDB	
	(DB 名記載無し)	Cassandra～MongoDB	
	(DB 名記載無し)	(DB 名以外の修正)	
3	Cassandra	NoSQL	誤りなし
	Cassandra	Cassandra/MongoDB	
	Cassandra	NoSQL	
	Cassandra	Cassandra/MongoDB	
4	Cassandra	NoSQL	誤りなし(キーワード未設定時は変更漏れ)
5	Cassandra	NoSQL	誤りなし(キーワード未設定時は変更漏れ)
6	Cassandra	Cassandra/MongoDB	誤りなし
	Cassandra	Cassandra～MongoDB	
	Cassandra	NoSQL	
	(DB 名記載無し)	Cassandra/MongoDB	
	(DB 名記載無し)	Cassandra～MongoDB	
7	Cassandra	Cassandra/MongoDB	誤りなし
8	Cassandra	Cassandra/MongoDB	誤りなし
9	Cassandra	NoSQL	誤りなし
	Cassandra	Cassandra	
	Cassandra	Cassandra	
	(DB 名記載無し)	MongoDB	
	(DB 名記載無し)	MongoDB	
10	Cassandra	Cassandra/MongoDB	誤りなし
11	Cassandra	Cassandra/MongoDB	誤りなし
	Cassandra	(DB 名以外の修正)	
12	Cassandra	Cassandra～MongoDB	誤りなし
	Cassandra	Cassandra/MongoDB	
13	Cassandra	Cassandra/MongoDB	誤りなし
	Cassandra	Cassandra/MongoDB	
14	Cassandra	Cassandra/MongoDB	誤りなし

6. 結論

本稿では、派生開発における課題として、変更箇所特定の精度・品質が開発者の能力によって異なることと、設計書変更後のレビュー漏れの 2 点を挙げ､解決策を検討した。解決策として、変更要求に応じた 3 つの抽出法であるキーワード検索・暗黙的参照・明示的参照を提案し、そのうちキーワード検索について実文書を用いて検証を行った。検証の結果、変更箇所候補の範囲が段落の場合に、全ての変更箇所を抽出できることを確認した。これにより、実文書の全ての内容を確認していた従来と比較し、確認範囲が 28.41%に削減される。今回、変更要求を限定し検証を行った。今後は、暗黙的参照・明示的参照に適している変更要求の検討を進めるとともに実文書を用いて検証を行う。

参考文献

[1]”派生開発推進協議会”. http://affordd.jp/(参照 2018-10-25)
[2]岸 知二. プロダクトライン開発の全体像と要求工学. 情報処理, 2009, Vol.50, No.4, p.268-273
[3]稗方 和夫, 大和 裕幸, 深田 直人, 中村 覚, 岡田 伊策, 齋藤 稔, 笈田 佳彰, 渡辺 郁雄, 松本 滋. システムの仕様変更調査における設計情報を用いた影響分析システムの開発. 第 24 回設計工学・システム部門講演会, 2014,
[4]位野木 万里, 大野 昭徳, 野村 典文. 要求仕様の一貫性検証支援ツールを用いた要求設計書のドキュメント品質の分析手法の提案と適用評価. 情報処理、2017-SE-195, No.6
[5]清水 吉男, 「派生開発」を成功させるプロセス改善の技術と極意. 技術評論社, 2007
[6]岸本 康成, 坂本 啓, 小林 透. ソースコードと設計書を用いたソフトウェアの派生関係の抽出. 情報処理, 2012, Vol.53 No.3, p.1137-1149